Department of Information Systems
Subject : Requirement Engineering

RE Process

IS184309, 3 sks

Dr. Rarasmaya Indraswari, S.Kom.

Outline

- ✓What's RE? (recap)
- ✓Why RE? (recap)
- ✓What are the objectives of RE?
- ✓RE Process & Techniques
- ✓Discussion

If you can’t describe what you are doing as a process, you don’t know what you’re doing.

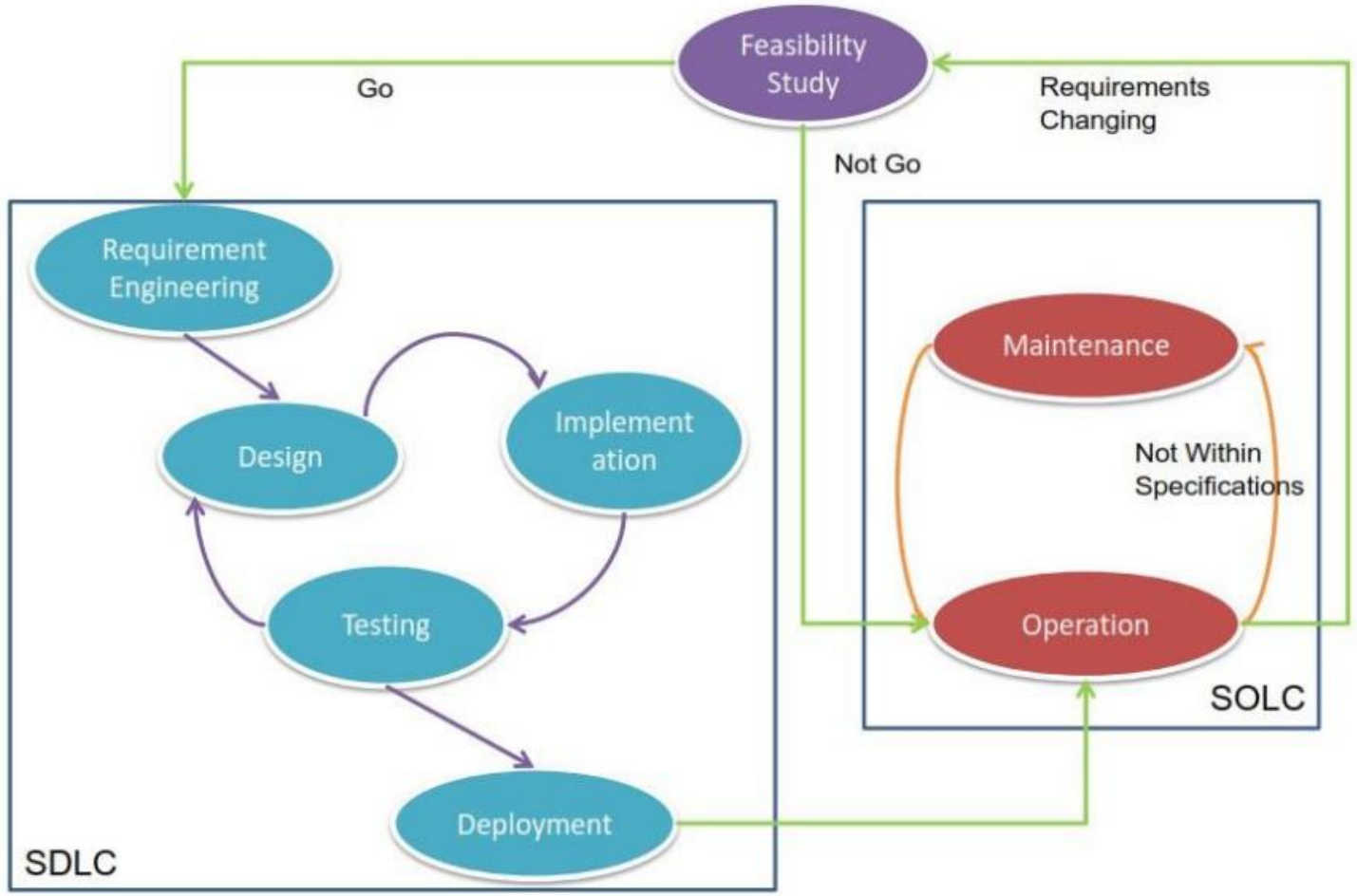
(William Edwards Deming, management consultant, 1900–93)

What is RE?

What is requirement engineering?

- **Requirements engineering** is the process of :
  - eliciting,
  - understanding,
  - specifying
  - and validating

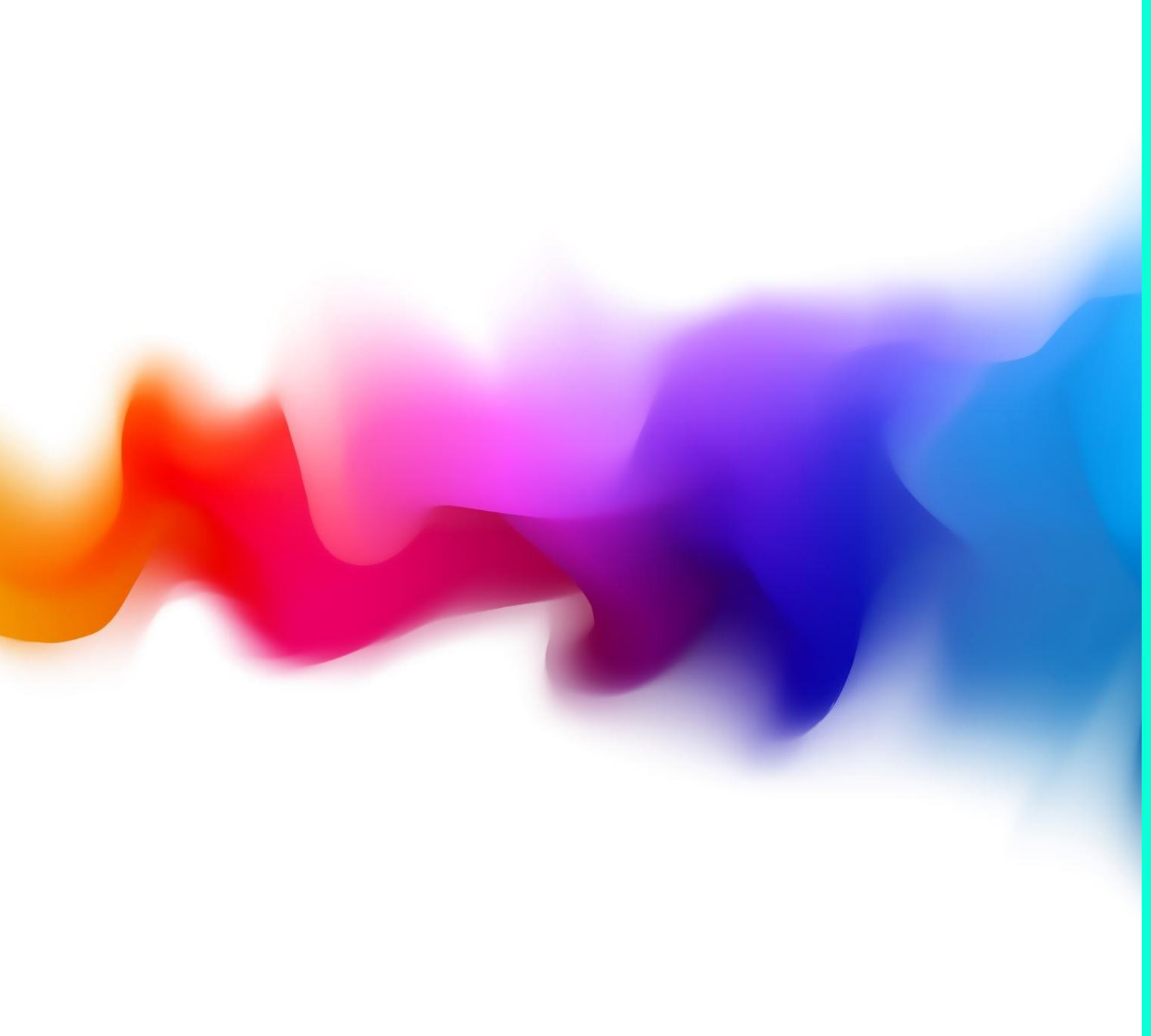
  customers' and **users**' **requirements**



Software Life Cycle

What is requirement engineering?

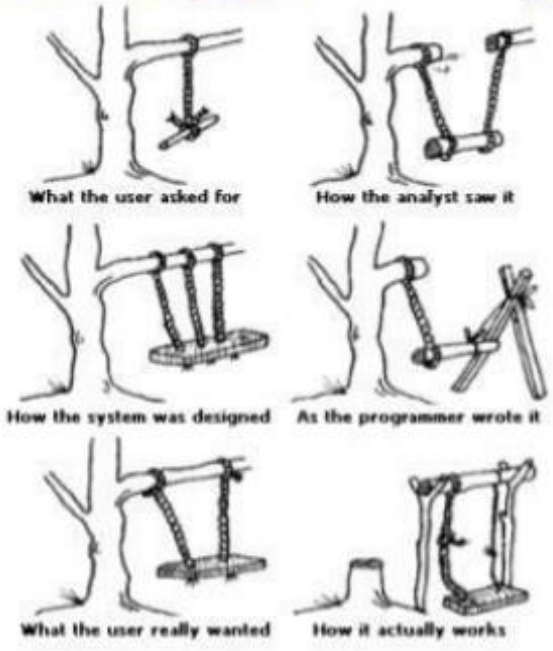
- **Requirement engineering** identifies technological restrictions under which the application should be constructed and run
- e.g. of technology restriction
  - Customer cannot ask a hard disk with unlimited capacity
  - Big data can impose a time consuming query in relational database
  - Traditional php engine can only process 250 MB data in memory



Why RE?

Requirements of Requirements

- **Clear**
- **Measurable**
- Feasible
- Necessary
- Prioritized
- Concise



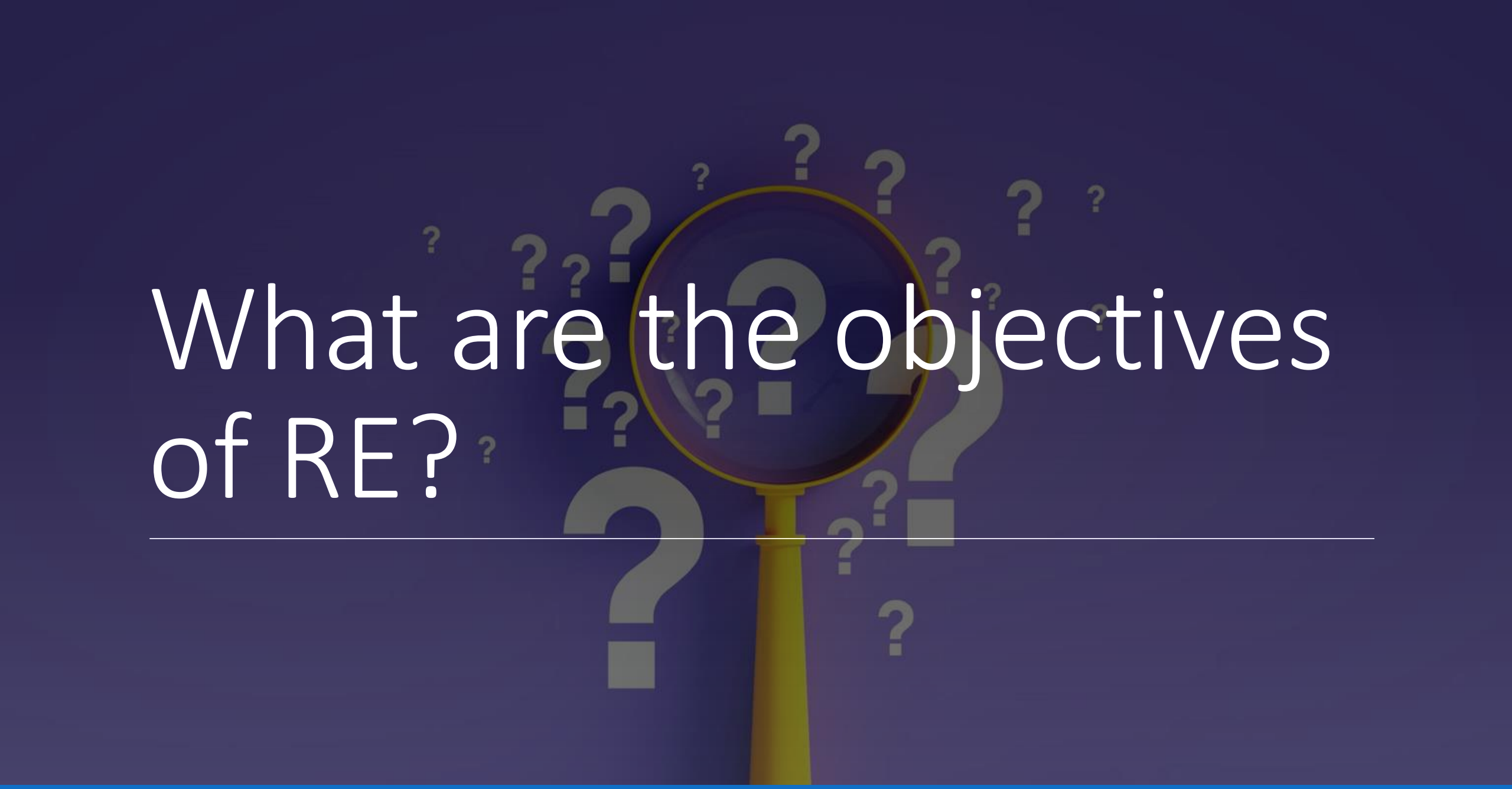
8

Designing and building an elegant **computer program that solves the wrong problem serves no one's needs**. That's why it's important to understand what the customer wants before we start to design and build a computer-based system

(Pressman, 2009)

Why RE?

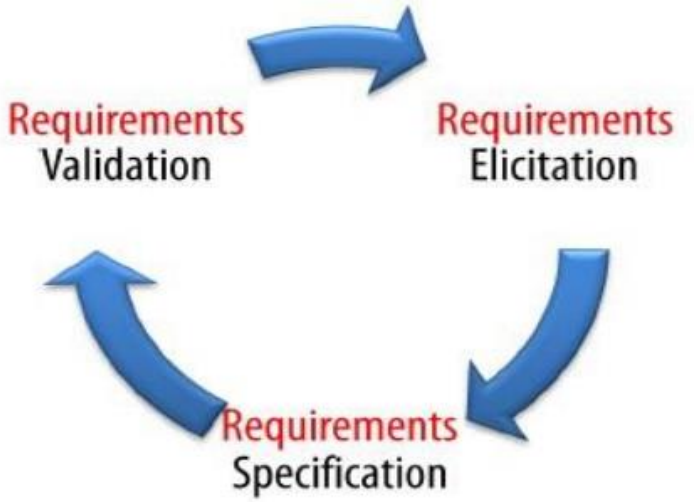
What are the objectives of RE?

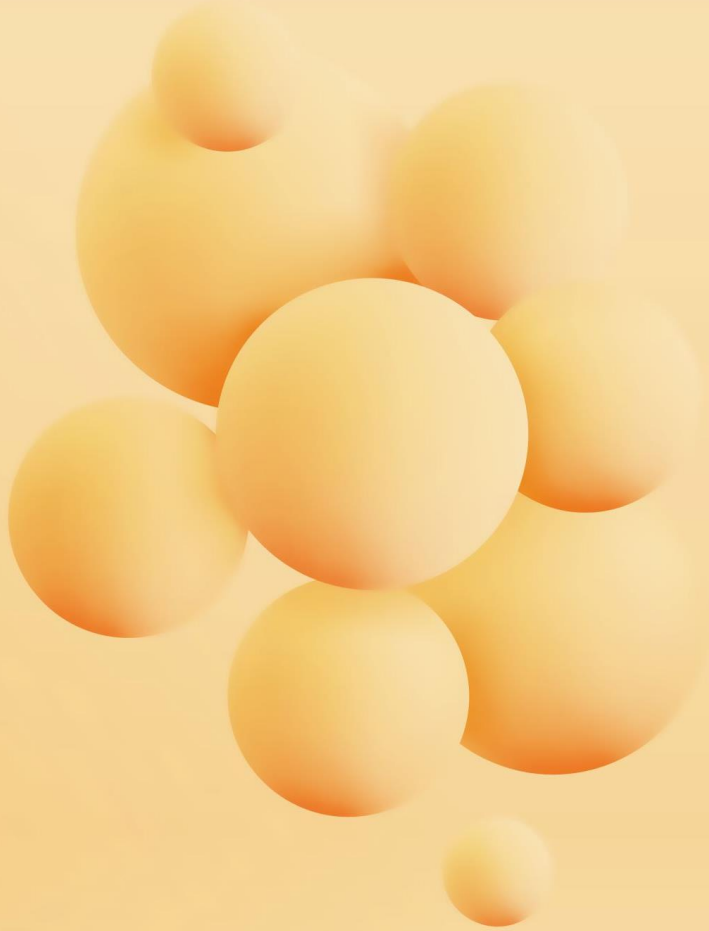


What are the objectives of RE?

- Analyse the problem
- Document the result in a variety of format
- Evaluate the precision of the results produced

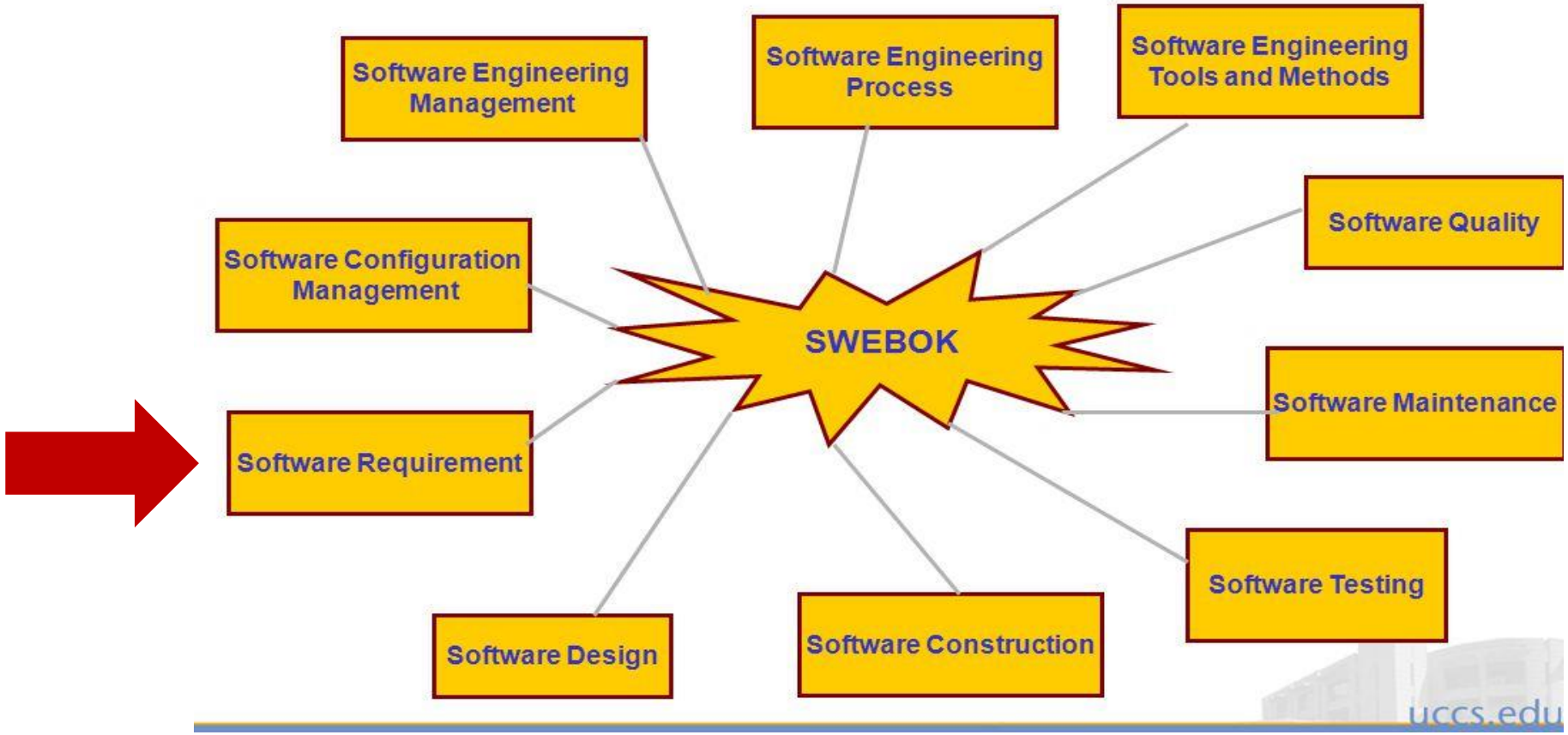
in **iterative ways.**



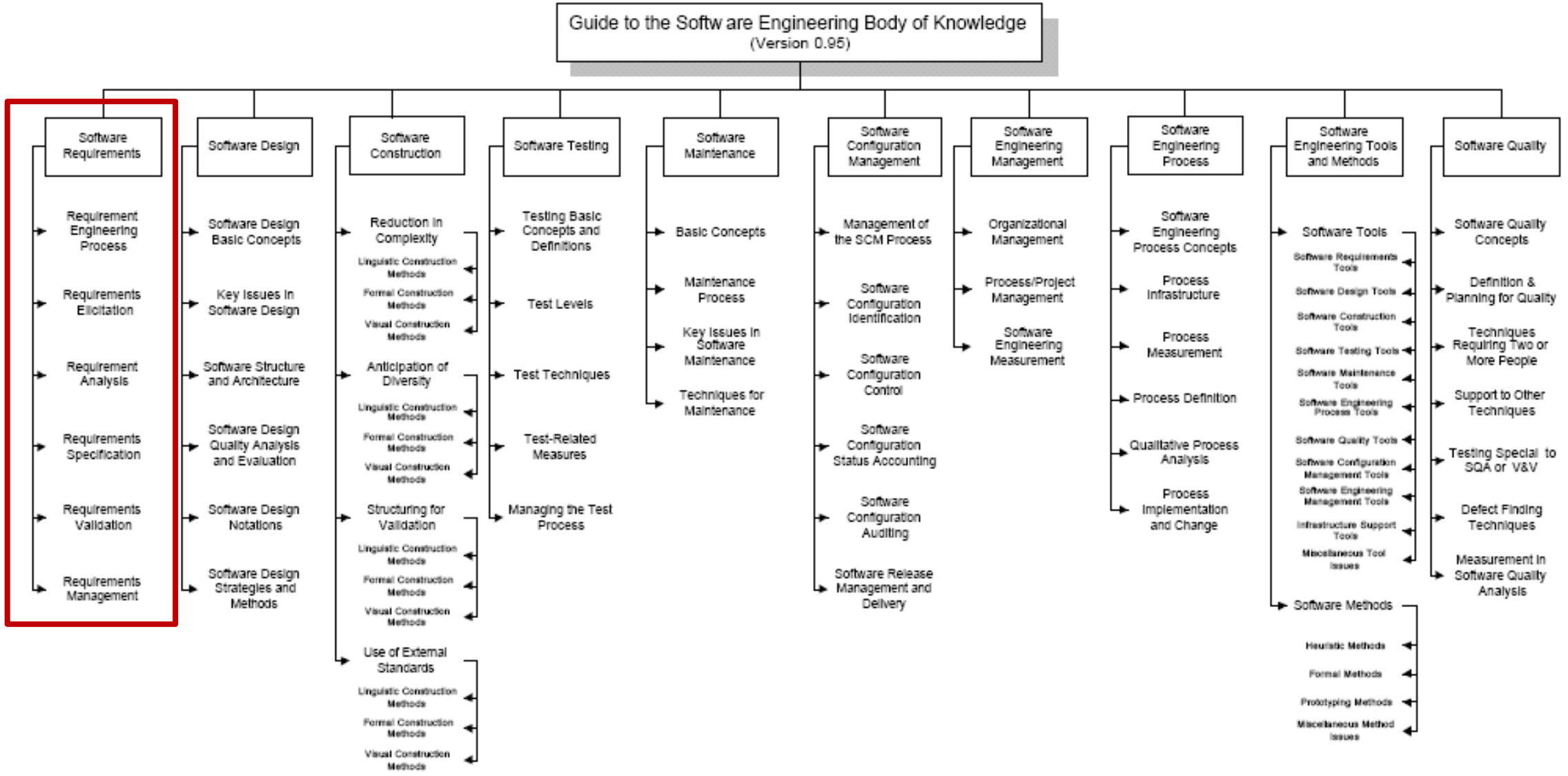


Requirement Engineering

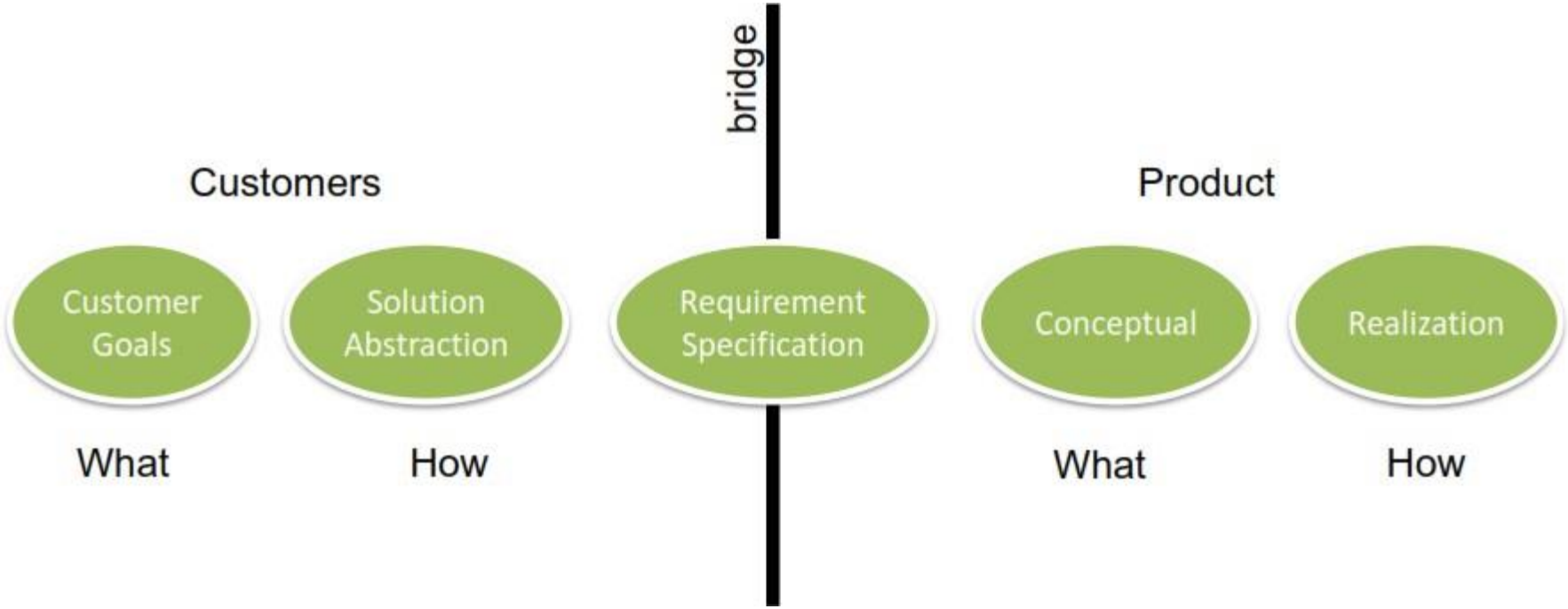
A complete SWEBOK based Education Program



SWEBOK


Guide to the Softw are Engineering Body of Knowledge
(Version 0.95)
Software Requirements
Requirement Engineering Process
Requirements Elicitation
Requirement Analysis
Requirements Specification
Requirements Validation
Requirements Management
Software Design
Software Design Basic Concepts
Key Issues In Software Design
Software Structure and Architecture
Software Design Quality Analysis and Evaluation
Software Design Notations
Software Design Strategies and Methods
Software Construction
Reduction In Complexity
Linguistic Construction Methods
Formal Construction Methods
Visual Construction Methods
Anticipation of Diversity
Linguistic Construction Methods
Formal Construction Methods
Visual Construction Methods
Structuring for Validation
Linguistic Construction Methods
Formal Construction Methods
Visual Construction Methods
Use of External Standards
Linguistic Construction Methods
Formal Construction Methods
Visual Construction Methods
Software Testing
Testing Basic Concepts and Definitions
Test Levels
Test Techniques
Test-Related Measures
Managing the Test Process
Software Maintenance
Basic Concepts
Maintenance Process
Key Issues In Software Maintenance
Techniques for Maintenance
Software Configuration Management
Management of the SCM Process
Software Configuration Identification
Software Configuration Control
Software Configuration Status Accounting
Software Configuration Auditing
Software Release Management and Delivery
Software Engineering Management
Organizational Management
Process/Project Management
Software Engineering Measurement
Software Engineering Process
Software Engineering Process Concepts
Process Infrastructure
Process Measurement
Process Definition
Qualitative Process Analysis
Process Implementation and Change
Software Engineering Tools and Methods
Software Tools
Software Requirements Tools
Software Design Tools
Software Construction Tools
Software Testing Tools
Software Maintenance Tools
Software Engineering Process Tools
Software Quality Tools
Software Configuration Management Tools
Software Engineering Management Tools
Infrastructure Support Tools
Miscellaneous Tool Issues
Software Methods
Heuristic Methods
Formal Methods
Prototyping Methods
Miscellaneous Method Issues
Software Quality
Software Quality Concepts
Definition & Planning for Quality
Techniques Requiring Two or More People
Support to Other Techniques
Testing Special to SQA or V&V
Defect Finding Techniques
Measurement in Software Quality Analysis

Focus of RE


bridge
Customers
Product
Customer Goals
Solution Abstraction
Requirement Specification
Conceptual
Realization
What
How
What
How

RE Process

RE Process

(Pressman & Maxin, 2014)

1. Inception

Ask a set of questions that establish …

- basic understanding of the **problem**
- the **people** who want a solution
- the nature of the **solution** that is desired, and
- the effectiveness of preliminary **communication** and **collaboration** between the customer and the developer

2. Elicitation

Elicit requirements from all stakeholders to **establish business goal** by asking some of the following questions to the users/customers:

- **objectives** for the system or product
- what is to be **accomplished**
- how the **system** or product **fits** into the needs of the business
- how the **system** or product is **to be used** on a day-to-day basis

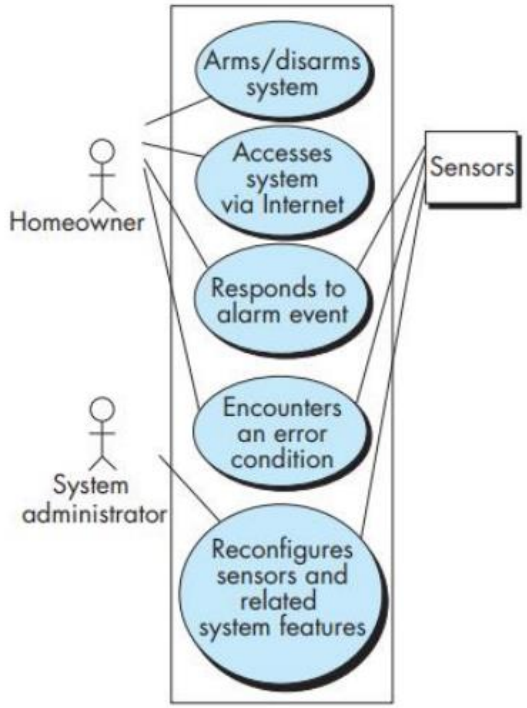
Output:

- A statement of need and **feasibility**
- A bounded statement of **scope** for the system or product
- A list of **stakeholders** in the requirements elicitation activity.
- A description of the **system's technical environment**.
- A **list of requirements** (function-based) and the **domain constraints** that apply to each.
- A set of **high-level usage scenarios**
- Any **prototypes** developed to better define requirements

2. Elicitation

Techniques:

1) Focus Group Discussion
2) Questioner
3) Brainstorming
4) Prototype
5) Observation
6) JAD – Joint Application Design
7) Goal-based Methods
8) Scenario-based Methods
9) Ethnography
10) Interview



High Level Use Case as one of the output (Pressman, 2014)

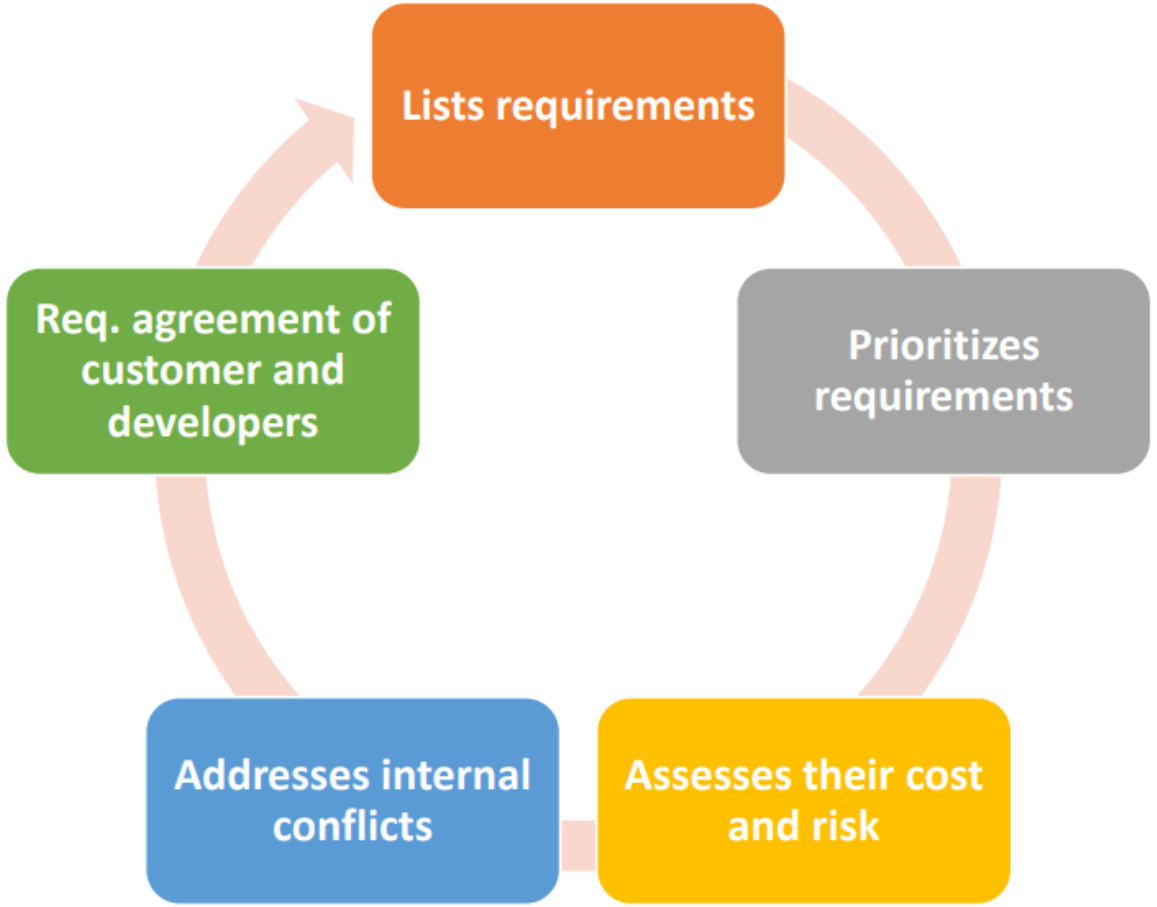
3. Elaboration

Create an analysis model that identifies **data, function and behavioral requirements**

- expands and refinement of **user scenarios** developed in inception phase that describe how the end user (and other actors) will interact with the system.
- The **relationships and collaboration between classes** are identified
- a variety of **supplementary diagrams** are produced

4. Analysis & Negotiation

Agree on a **deliverable system** that is **realistic** for **developers** and **customers**

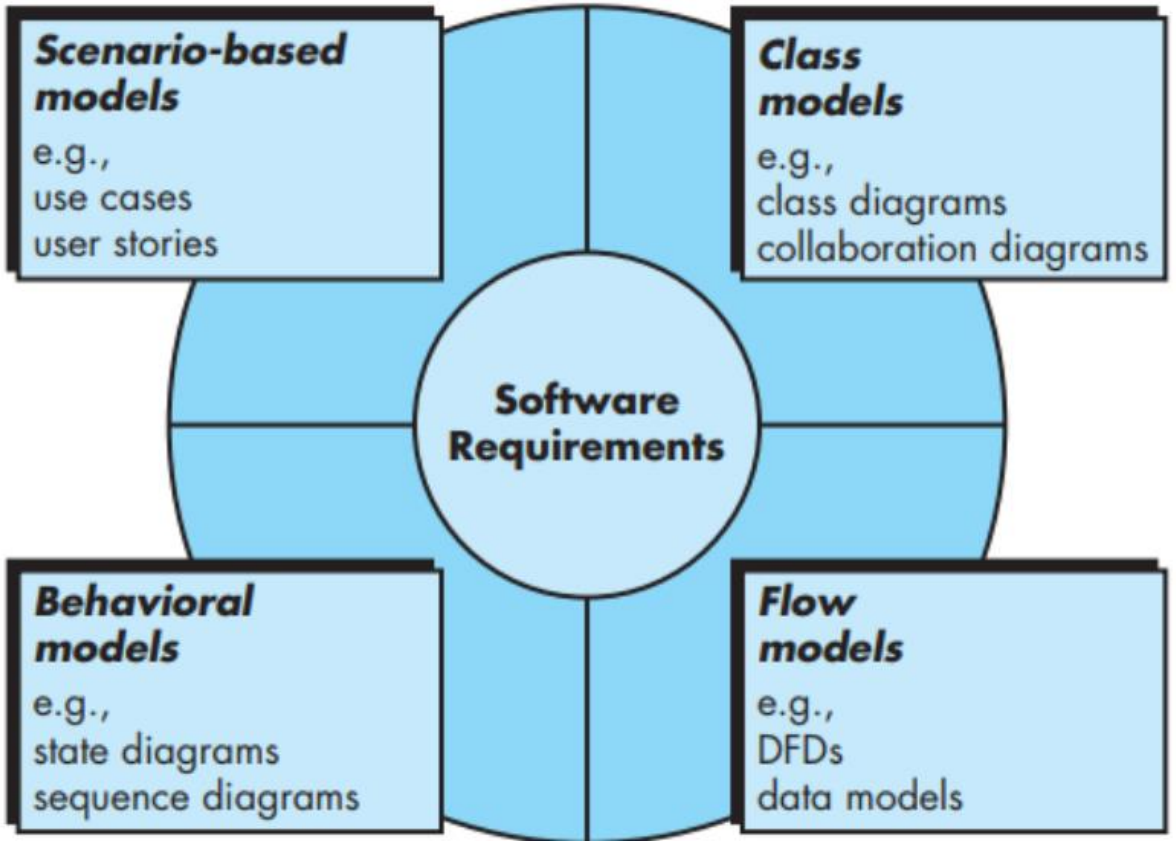


5. Specification & System Modelling

Can be any one (or more) of the following:

- A **written document**
- A set of **models**
- A formal **mathematical**
- A collection of **user scenarios** (use-cases)
- A **prototype**

5. Specification & System Modelling



Requirements modeling action results in one or more of the following types of models:

- **Scenario-based** (various actors-based)
- ***Class-oriented models*** that represent object-oriented classes to achieve system requirements.
- ***Behavioral and patterns-based*** *models* that depict how the software behaves as a consequence of external "events."
- ***Data models*** that depict the information domain for the problem.
- ***Flow-oriented models*** that represent the functional elements of the system and how they transform data as they move through the system

SafeHome

Use Case Template for Surveillance

Use case: Access camera surveillance via the Internet—display camera views (ACS-DCV)

Iteration: 2, last modification: January 14 by V. Raman.

Primary actor: Homeowner.

Goal in context: To view output of camera placed throughout the house from any remote location via the Internet.

Preconditions: System must be fully configured; appropriate user ID and passwords must be obtained.

Trigger: The homeowner decides to take a look inside the house while away.

Scenario:

1. The homeowner logs onto the *SafeHome Products* website.
2. The homeowner enters his or her user ID.
3. The homeowner enters two passwords (each at least eight characters in length).
4. The system displays all major function buttons.
5. The homeowner selects the "surveillance" from the major function buttons.
6. The homeowner selects "pick a camera."
7. The system displays the floor plan of the house.
8. The homeowner selects a camera icon from the floor plan.
9. The homeowner selects the "view" button.
10. The system displays a viewing window that is identified by the camera ID.
11. The system displays video output within the viewing window at one frame per second.

Exceptions:

1. ID or passwords are incorrect or not recognized—see use case **Validate ID and passwords.**
2. Surveillance function not configured for this system—system displays appropriate error message; see use case **Configure surveillance function.**
3. Homeowner selects "View thumbnail snapshots for all camera"—see use case **View thumbnail snapshots for all cameras.**
4. A floor plan is not available or has not been configured—display appropriate error message and see use case **Configure floor plan.**
5. An alarm condition is encountered—see use case **alarm condition encountered.**

Priority: Moderate priority, to be implemented after basic functions.

When available: Third increment.

Frequency of use: Infrequent.

Channel to actor: Via PC-based browser and Internet connection.

Secondary actors: System administrator, cameras.

Channels to secondary actors:

1. System administrator: PC-based system.
2. Cameras: wireless connectivity.

Open issues:

1. What mechanisms protect unauthorized use of this capability by employees of *SafeHome Products*?
2. Is security sufficient? Hacking into this feature would represent a major invasion of privacy.
3. Will system response via the Internet be acceptable given the bandwidth required for camera views?
4. Will we develop a capability to provide video at a higher frames-per-second rate when high-bandwidth connections are available?

5. Specification & System Modelling

EXAMPLE OF DETAIL USE CASE SCENARIO TEMPLATE (PRESSMAN, 2014)

6. Validation

Illustration example:

- **The software should be user friendly.**

  This is too vague for developers to test or assess. What exactly does “user friendly” mean? To validate it, it must be quantified or qualified in some manner.

- **The probability of a successful unauthorized database intrusion should be less than 0.0001.**

  The second requirement has a quantitative element (“less than 0.0001”), but intrusion testing will be difficult and time consuming.
  Is this level of security even warranted for the application? Can other complementary requirements associated with security (e.g., password protection, specialized handshaking) replace the quantitative requirement noted?

6. Validation

a **review mechanism** that looks for

- **errors** in content or interpretation
- **areas** where **clarification** may be required
- **missing information**
- **inconsistencies** (a major problem when large products or systems are engineered)
- **conflicting** or **unrealistic** (unachievable) requirements.

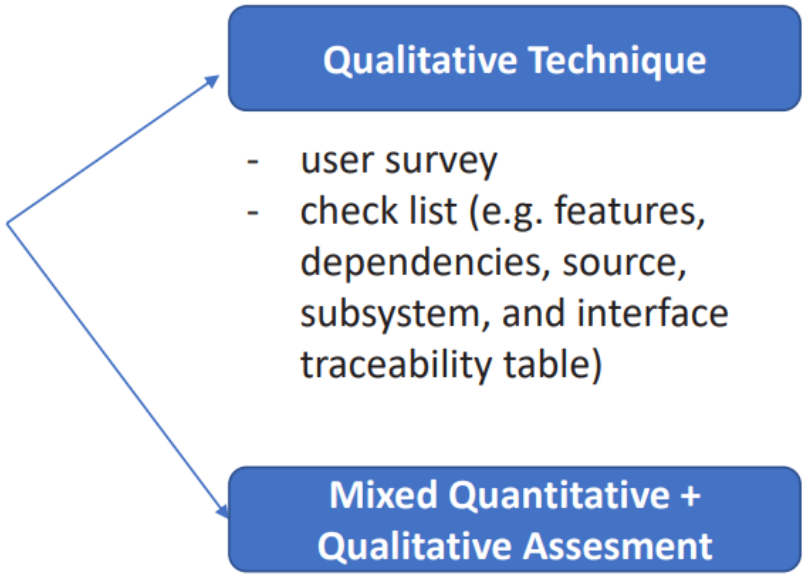
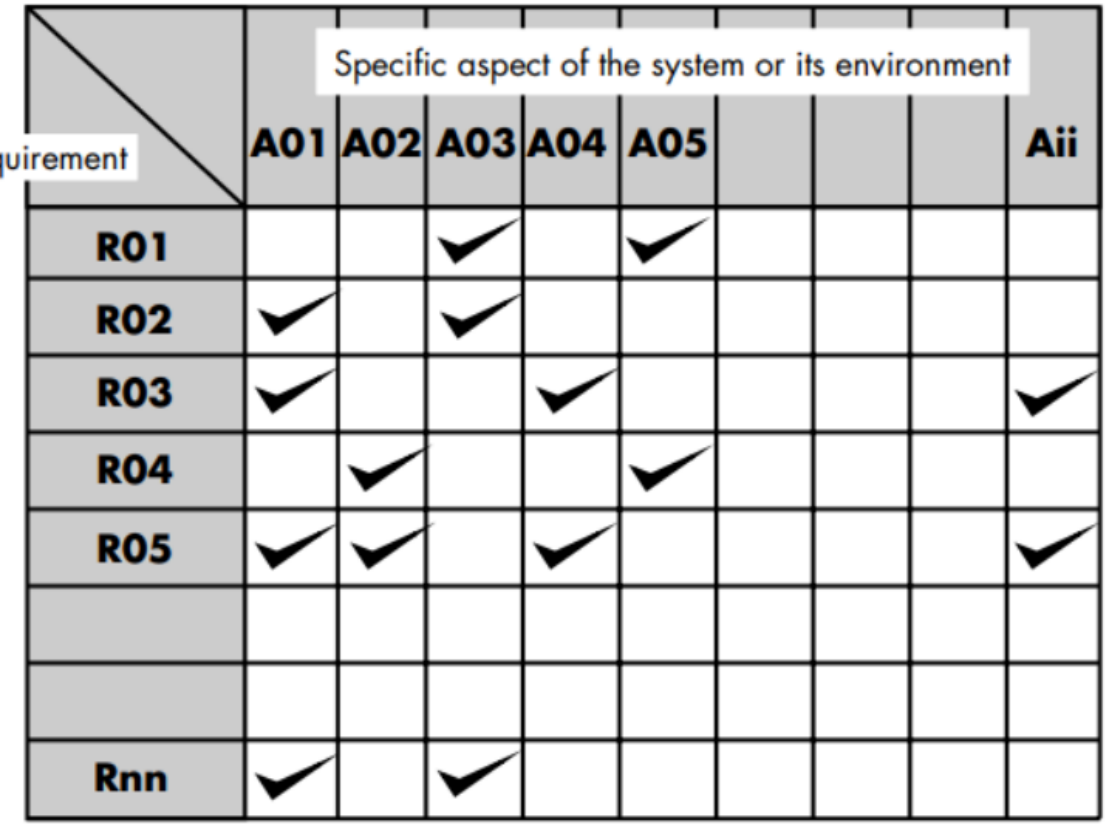


FIGURE 10.4
Generic traceability table



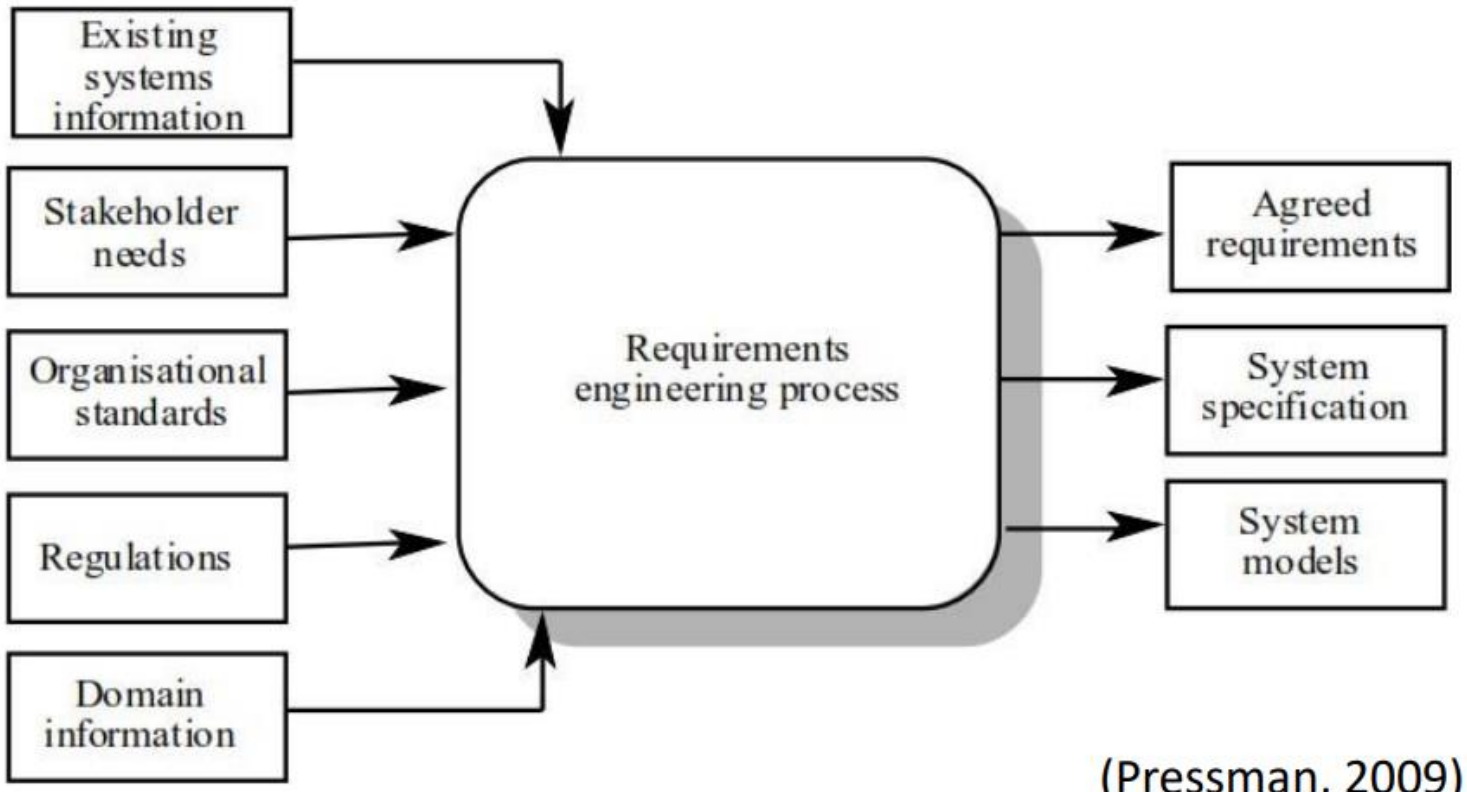
Example of traceability table (Pressman, 2009)

6. Validation

Set of activities that help the project team **identify**, **control**, and **track** **requirements** and **changes** to requirements at **any time** as the project proceeds

7. Requirement Management

RE Process (Input & Output)



(Pressman, 2009)

Task 3

1. Create a group of 4-5 students. There should be a total of 10 groups in a class.
2. Each group should make presentation (.pptx) regarding the corresponding topic number:
    1. Functional Requirement (FR)
    2. Non-functional Requirement (NFR)
    3. Business Use-Case
    4. Use-Case Diagram
    5. Use-Case Scenario
    6. SMART Requirement: Specific
    7. SMART Requirement: Measurable
    8. SMART Requirement: Attainable
    9. SMART Requirement: Realizable
    10. SMART Requirement: Time-Bounded / Traceable
3. Give an example for your topic.
4. Submit your presentation file (.pptx) in the Classroom. Each group will present their topic on the class between week 4-6.
5. The length / number of presentation slide is not limited.